DIGITAL SYSTEM DESIGN LABORATORY

LAB 4

BASIC BUILDING BLOCKS OF SINGLE CYCLE MICROPROCESSOR

I. LAB OBJECTIVES

This Lab experiments are intended to implement basic building blocks of Single Cycle Microprocessor

II. DESCRIPTION

Single Cycle Microprocessor datapath to be implemented is in figure 2.1.

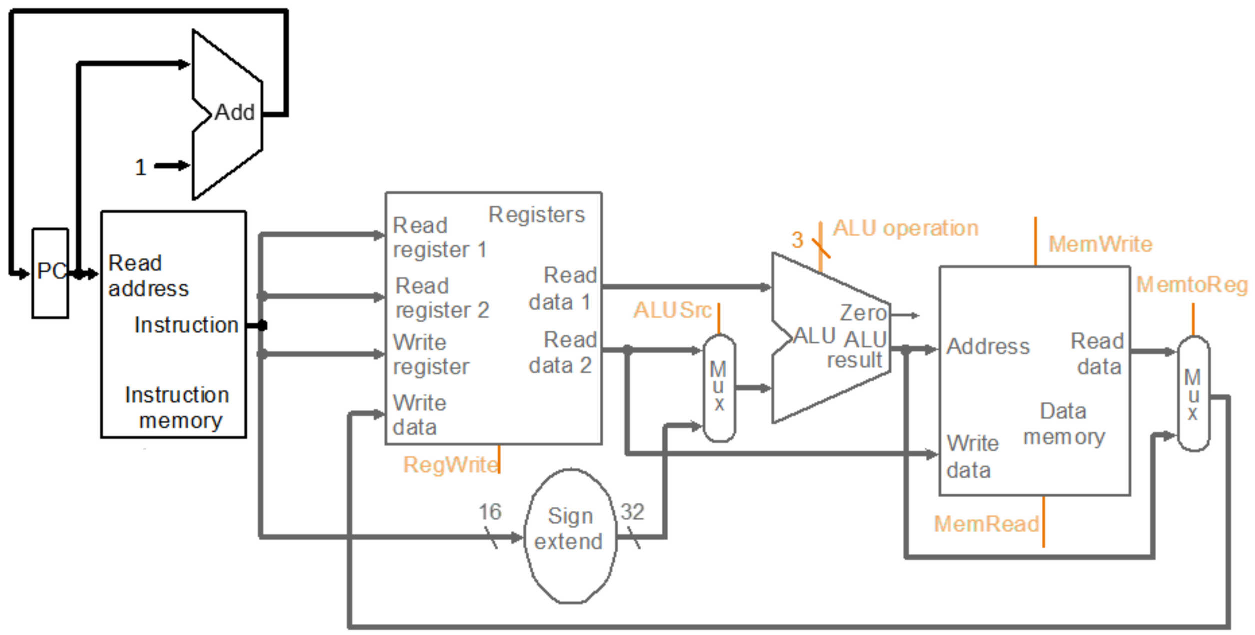
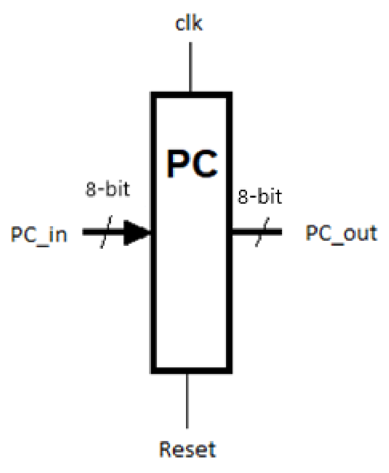


Figure 2.1: Single Cycle Microprocessor DataPath

III. LAB PROCEDURE

III.1 EXPERIMENT NO. 1

III.1.1 AIM: To implement Program Counter

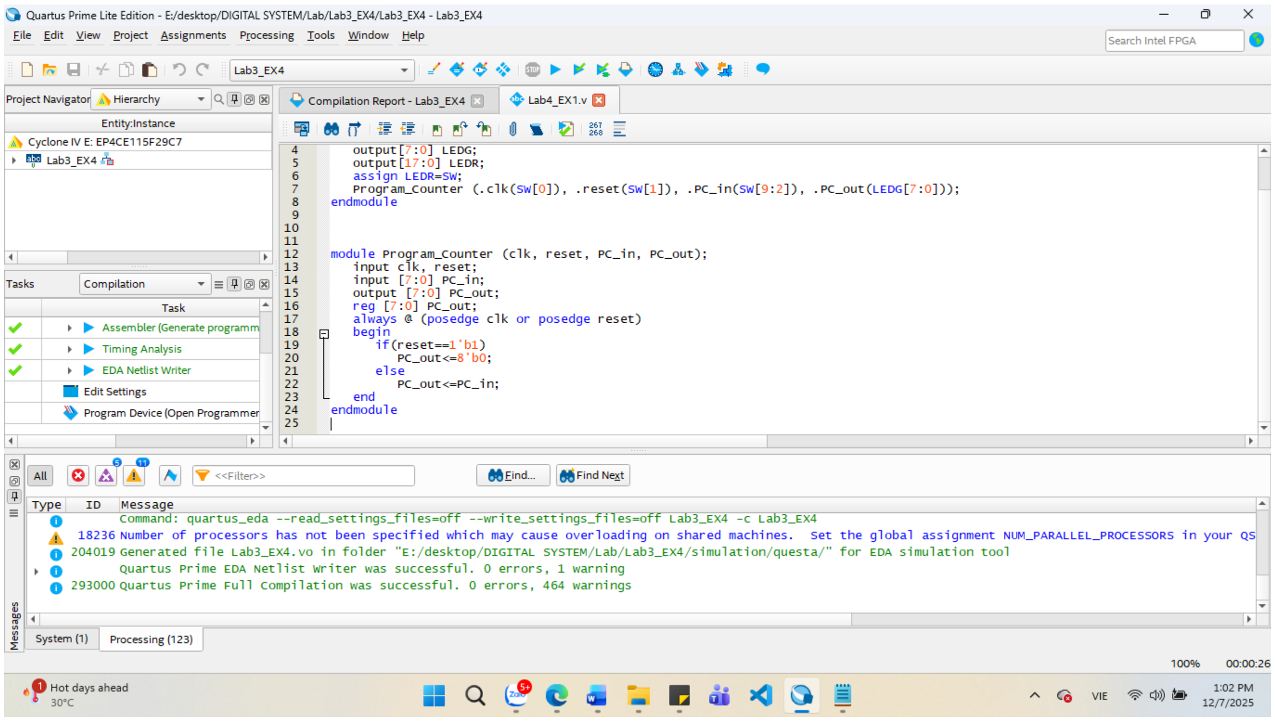


III.1.2 CODE

```
module Program_Counter (clk, reset, PC_in, PC_out);
        input clk, reset;
        input [7:0] PC_in;
        output [7:0] PC_out;
        reg [7:0] PC_out;
        always @ (posedge clk or posedge reset)
        begin
                if(reset==1'b1)
                        PC_out<=8’b0;
                else
                        PC_out<=PC_in;
        end
endmodule
```

III.1.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT



```
    output[7:0] LEDG;
    output[17:0] LEDR;
    assign LEDR=SW;
    Program_Counter (.clk(SW[0]), .reset(SW[1]), .PC_in(SW[9:2]), .PC_out(LEDG[7:0]));
endmodule



module Program_Counter (clk, reset, PC_in, PC_out);
    input clk, reset;
    input [7:0] PC_in;
    output [7:0] PC_out;
    reg [7:0] PC_out;
    always @ (posedge clk or posedge reset)
    begin
       if(reset==1'b1)
          PC_out<=8'b0;
       else
          PC_out<=PC_in;
    end
endmodule
```

```
Type  ID     Message
             Command: quartus_eda --read_settings_files=off --write_settings_files=off Lab3_EX4 -c Lab3_EX4
      18236  Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your QS
      204019 Generated file Lab3_EX4.vo in folder "E:/desktop/DIGITAL SYSTEM/Lab/Lab3_EX4/simulation/questa/" for EDA simulation tool
             Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 464 warnings
```

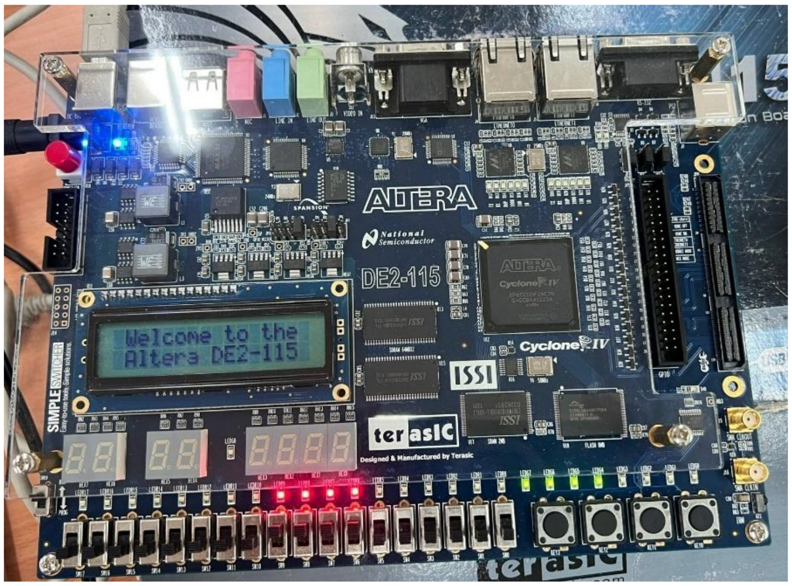
Quartus Prime Lite Edition - E:/desktop/DIGITAL SYSTEM/Lab/Lab3_EX4/Lab3_EX4 - Lab3_EX4
File Edit View Project Assignments Processing Tools Window Help
Lab3_EX4
Compilation Report - Lab3_EX4
Lab4_EX1.v
Cyclone IV E: EP4CE115F29C7
Lab3_EX4
Assembler (Generate programm
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer
System (1)
Processing (123)

EX: reset = 1 → PC_out = 00000000

LEDG7	LEDG6	LEDG5	LEDG4	LEDG3	LEDG2	LEDG1	LEDG0
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

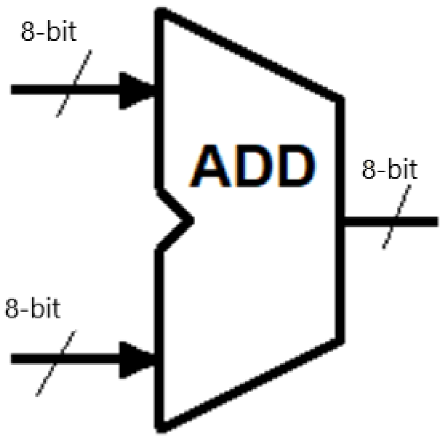
Reset = 0 → PC_out =PC_in = 8'b11110000

LEDG7	LEDG6	LEDG5	LEDG4	LEDG3	LEDG2	LEDG1	LEDG0
ON	ON	ON	ON	OFF	OFF	OFF	OFF



III.2 EXPERIMENT NO. 2

III.2.1 AIM: To implement 32 bit Adder

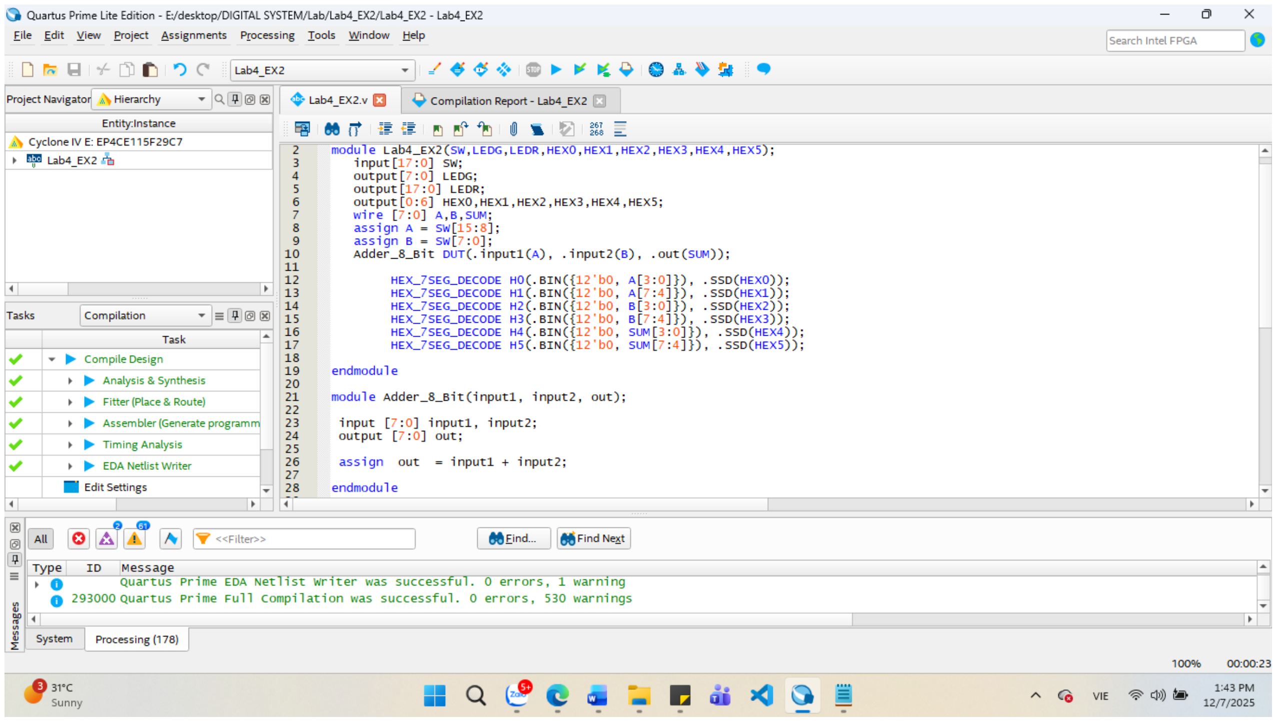


III.2.2 CODE

```
module Adder32Bit(input1, input2, out);

        input [7:0] input1, input2;
        output [7:0] out;
        reg [7:0]out;
        always@( input1 or input2)
        begin
                out  <= input1 + input2;
        end
endmodule
```

III.2.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT


Quartus Prime Lite Edition - E:/desktop/DIGITAL SYSTEM/Lab/Lab4_EX2/Lab4_EX2 - Lab4_EX2
File Edit View Project Assignments Processing Tools Window Help
Lab4_EX2
Project Navigator Hierarchy
Entity:Instance
Cyclone IV E: EP4CE115F29C7
Lab4_EX2
Lab4_EX2.v
Compilation Report - Lab4_EX2
Tasks Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
Timing Analysis
EDA Netlist Writer
Edit Settings
All
Find...
Find Next
System
Processing (178)
100% 00:00:23
31°C Sunny
VIE
1:43 PM 12/7/2025

```verilog
module Lab4_EX2(SW,LEDG,LEDR,HEX0,HEX1,HEX2,HEX3,HEX4,HEX5);
   input[17:0] SW;
   output[7:0] LEDG;
   output[17:0] LEDR;
   output[0:6] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5;
   wire [7:0] A,B,SUM;
   assign A = SW[15:8];
   assign B = SW[7:0];
   Adder_8_Bit DUT(.input1(A), .input2(B), .out(SUM));

        HEX_7SEG_DECODE H0(.BIN({12'b0, A[3:0]}), .SSD(HEX0));
        HEX_7SEG_DECODE H1(.BIN({12'b0, A[7:4]}), .SSD(HEX1));
        HEX_7SEG_DECODE H2(.BIN({12'b0, B[3:0]}), .SSD(HEX2));
        HEX_7SEG_DECODE H3(.BIN({12'b0, B[7:4]}), .SSD(HEX3));
        HEX_7SEG_DECODE H4(.BIN({12'b0, SUM[3:0]}), .SSD(HEX4));
        HEX_7SEG_DECODE H5(.BIN({12'b0, SUM[7:4]}), .SSD(HEX5));

endmodule

module Adder_8_Bit(input1, input2, out);

 input [7:0] input1, input2;
 output [7:0] out;

 assign  out  = input1 + input2;

endmodule
```

```
Type  ID     Message
             Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 530 warnings
```

EX
A[15:8] = 0001 1000 (0x18 = 24)
B[7:0] = 0000 0011 (0x03 = 3)
SUM   = 0001 1011 (0x1B=27)

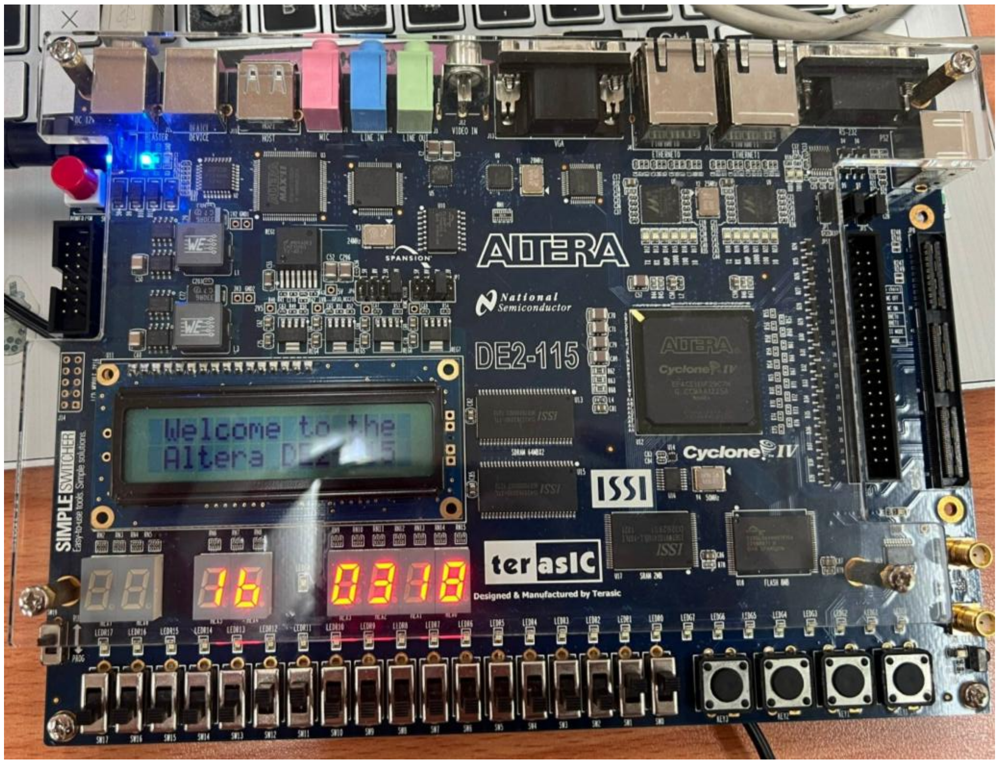
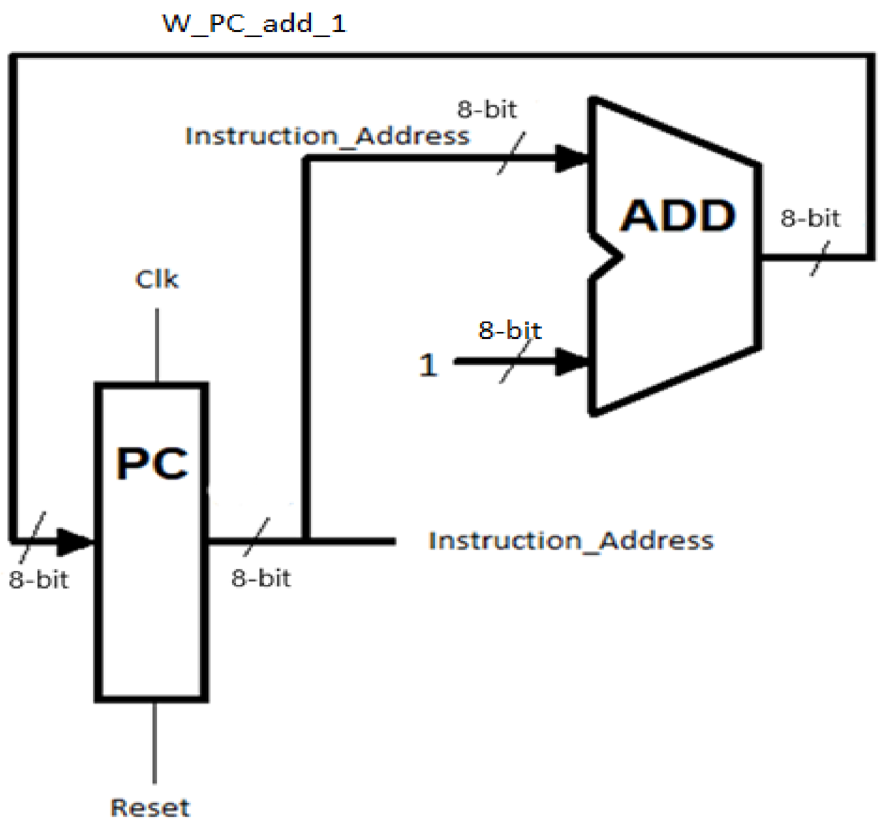


Figure 1 Test lại Kit

III.3 EXPERIMENT NO. 3

III.3.1 AIM: To implement the datapath for PC = PC + 1

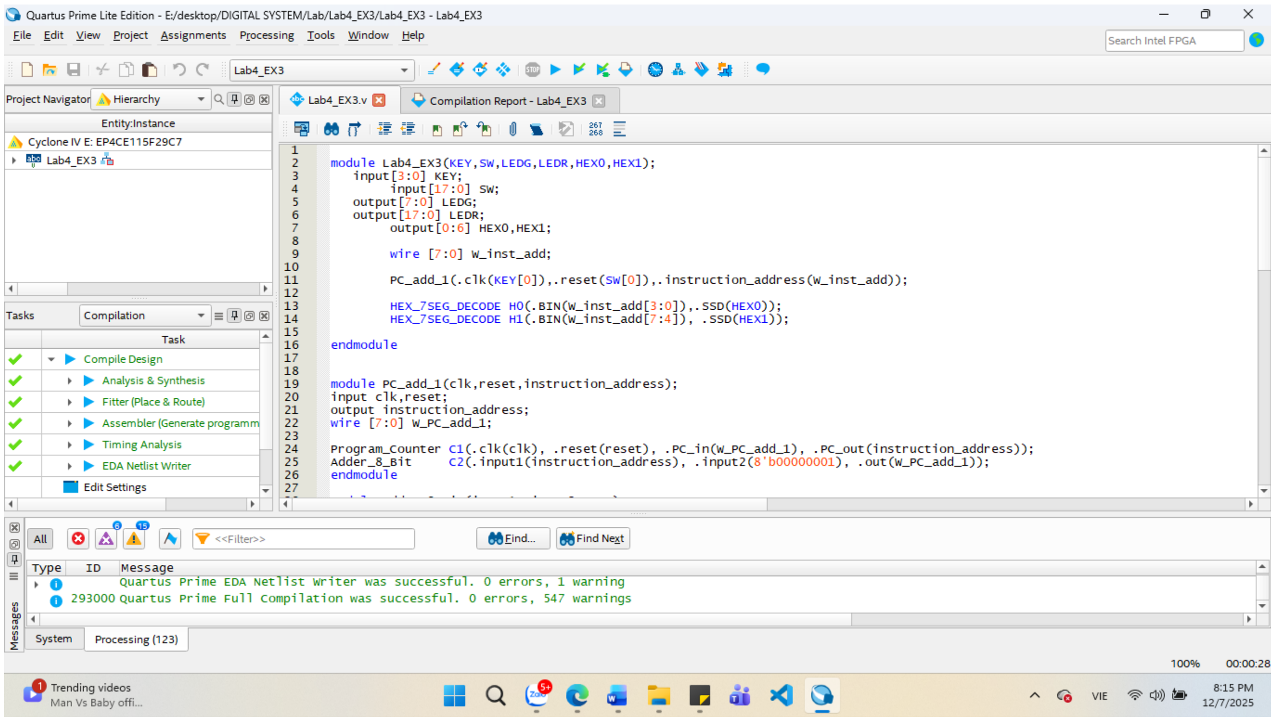


III.3.2 CODE

```
module PC_add_1 (clk, reset, Instruction_Address);
        input clk, reset;
        output [7:0] Instruction_Address;

//Your Verilog code here


endmodule
```

III.3.3 LAB ASSIGNMENT

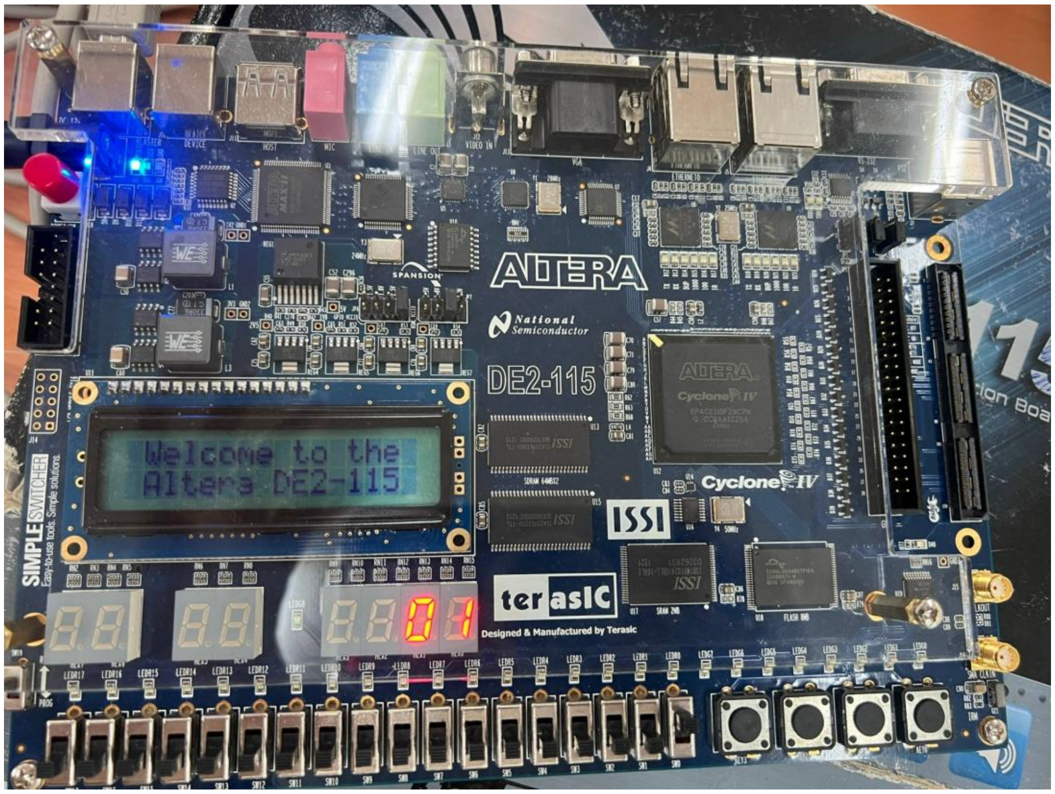
1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT


Quartus Prime Lite Edition - E:/desktop/DIGITAL SYSTEM/Lab/Lab4_EX3/Lab4_EX3 - Lab4_EX3
File Edit View Project Assignments Processing Tools Window Help
Lab4_EX3.v
Compilation Report - Lab4_EX3
Cyclone IV E: EP4CE115F29C7
Lab4_EX3
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
Timing Analysis
EDA Netlist Writer
Edit Settings

```
module Lab4_EX3(KEY,SW,LEDG,LEDR,HEX0,HEX1);
   input[3:0] KEY;
         input[17:0] SW;
   output[7:0] LEDG;
   output[17:0] LEDR;
         output[0:6] HEX0,HEX1;

         wire [7:0] W_inst_add;

         PC_add_1(.clk(KEY[0]),.reset(SW[0]),.instruction_address(W_inst_add));

         HEX_7SEG_DECODE H0(.BIN(W_inst_add[3:0]),.SSD(HEX0));
         HEX_7SEG_DECODE H1(.BIN(W_inst_add[7:4]), .SSD(HEX1));

endmodule


module PC_add_1(clk,reset,instruction_address);
input clk,reset;
output instruction_address;
wire [7:0] W_PC_add_1;

Program_Counter C1(.clk(clk), .reset(reset), .PC_in(W_PC_add_1), .PC_out(instruction_address));
Adder_8_Bit     C2(.input1(instruction_address), .input2(8'b00000001), .out(W_PC_add_1));
endmodule
```

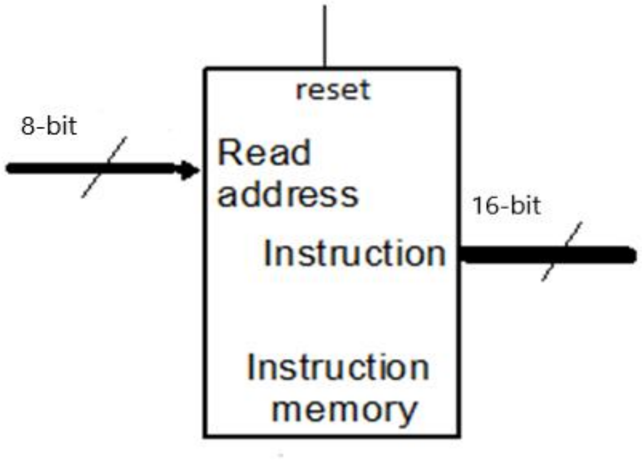
```
Type  ID     Message
             Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 547 warnings
```

Input = LED[7:0] = 0000 0001
KEY[0] = 1 → HEX = 01
KEY[0] = 10 → HEX = 0A



III.4 EXPERIMENT NO. 4

III.4.1 AIM: To implement the Instruction memory



III.4.2 CODE

```
module Instruction_Memory (read_address, instruction, reset);
        input reset;
        input [9:0] read_address;
        output [15:0] instruction;
        reg [15:0] Imemory [1024:0];

        Imemory[0] = 32'b0010000000001000 0000000000100000;
        //addi $t0, $zero, 32
        Imemory[00000001] = 32'b0010000000001001 0000000000110111;
        //addi $t1, $zero, 55
        Imemory[2] = 32'b00000001000010011000000000100100;
        //and $s0, $t0, $t1
        Imemory[3] = 32'b00000001000010011000000000100101;
        //or $s0, $t0, $t1
        Imemory[4] = 32'b10101100000100000000000000000100;
        //sw $s0, 4($zero)
        Imemory[5] = 32'b10101100000010000000000000001000;
        //sw $t0, 8($zero)
        Imemory[6] = 32'b00000001000010011000100000100000;
        //add $s1, $t0, $t1
        Imemory[7] = 32'b00000001000010011001000000100010;
        //sub $s2, $t0, $t1
        Imemory[8] = 32'b00010010001100100000000000001001;
        //beq $s1, $s2, error0
        Imemory[9] = 32'b10001100000100010000000000000100;
```

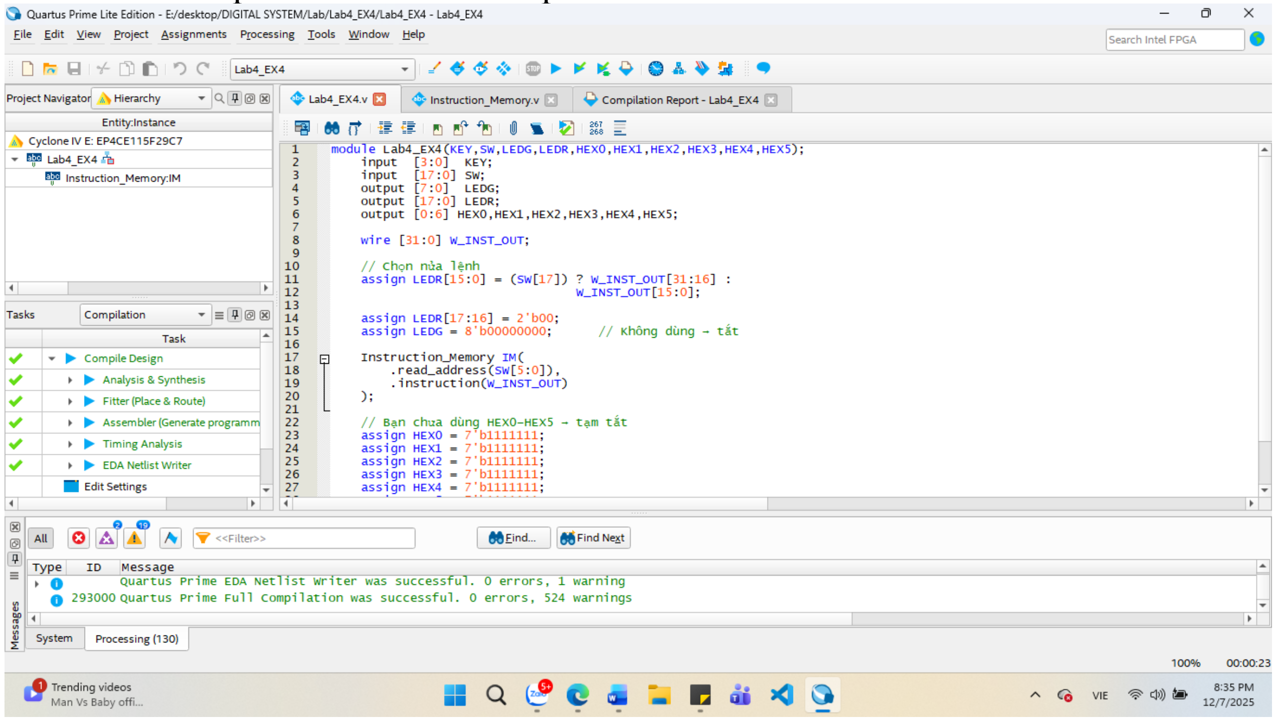
```
//lw $s1, 4($zero)
Imemory[10]= 32'b00110010001100100000000001001000;
//andi $s2, $s1, 48
Imemory[11] =32'b00010010001100100000000000001001;
//beq $s1, $s2, error1
Imemory[12] =32'b10001100000100110000000000001000;
//lw $s3, 8($zero)
Imemory[13] =32'b00010010000100110000000000001010;
//beq $s0, $s3, error2
Imemory[14] =32'b00000010010100011010000000101010;
//slt $s4, $s2, $s1 (Last)
Imemory[15] =32'b00010010100000000000000000001111;
//beq $s4, $0, EXIT
Imemory[16] =32'b00000010001000001001000000100000;
//add $s2, $s1, $0
Imemory[17] =32'b00001000000000000000000000001110;
//j Last
Imemory[18] =32'b00100000000010000000000000000000;
//addi $t0, $0, 0(error0)
Imemory[19] =32'b00100000000010010000000000000000;
//addi $t1, $0, 0
Imemory[20] =32'b00001000000000000000000000011111;
//j EXIT
Imemory[21] =32'b00100000000010000000000000000001;
//addi $t0, $0, 1(error1)
Imemory[22] =32'b00100000000010010000000000000001;
//addi $t1, $0, 1
Imemory[23] =32'b00001000000000000000000000011111;
//j EXIT
Imemory[24] =32'b00100000000010000000000000000010;
//addi $t0, $0, 2(error2)
Imemory[25] =32'b00100000000010010000000000000010;
//addi $t1, $0, 2
Imemory[26] =32'b00001000000000000000000000011111;
//j EXIT
Imemory[27] =32'b00100000000010000000000000000011;
//addi $t0, $0, 3(error3)
Imemory[28] =32'b00100000000010010000000000000011;
//addi $t1, $0, 3
Imemory[29] =32'b00001000000000000000000000011111;
//j EXIT
```

```
        end
endmodule
```

reset	Read_address	PC_out
1	32’b0	32'b00100000000010000000000000100000
0	32’b0	32'b00100000000010000000000000100000
0	32’b0000….. 0011	32'b00000001000010011000000000100101
0	32’b0000 …….0111	32'b00000001000010011001000000100010

III.4.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT


Quartus Prime Lite Edition - E:/desktop/DIGITAL SYSTEM/Lab/Lab4_EX4/Lab4_EX4 - Lab4_EX4

Project Navigator: Hierarchy — Entity:Instance
- Cyclone IV E: EP4CE115F29C7
- Lab4_EX4
  - Instruction_Memory:IM

Tasks: Compilation
- ✓ Compile Design
- ✓ Analysis & Synthesis
- ✓ Fitter (Place & Route)
- ✓ Assembler (Generate programm
- ✓ Timing Analysis
- ✓ EDA Netlist Writer
- Edit Settings

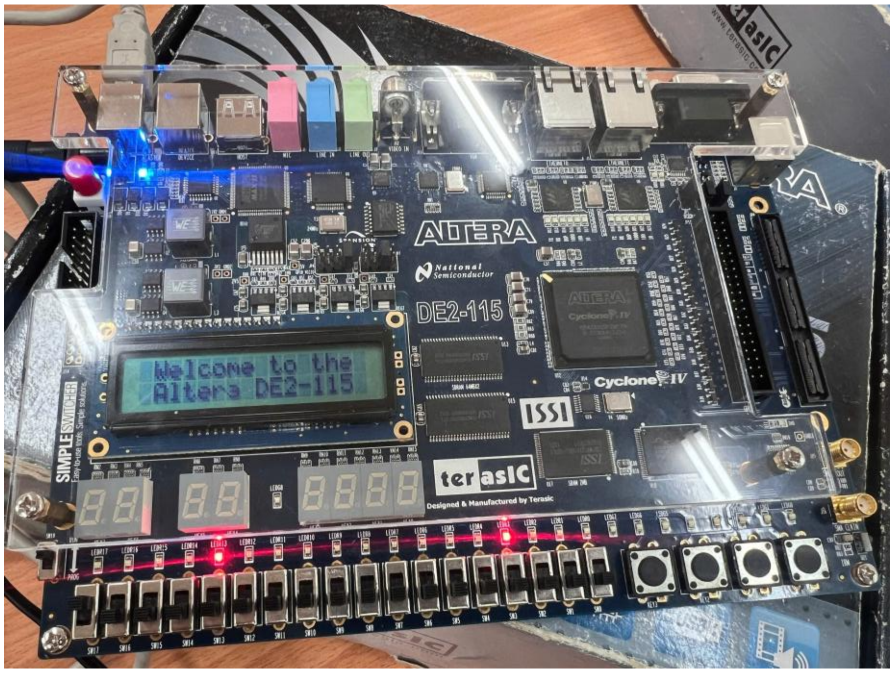
Lab4_EX4.v

```
module Lab4_EX4(KEY,SW,LEDG,LEDR,HEX0,HEX1,HEX2,HEX3,HEX4,HEX5);
    input  [3:0]  KEY;
    input  [17:0] SW;
    output [7:0]  LEDG;
    output [17:0] LEDR;
    output [0:6] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5;

    wire [31:0] W_INST_OUT;

    // Chọn nửa lệnh
    assign LEDR[15:0] = (SW[17]) ? W_INST_OUT[31:16] :
                                   W_INST_OUT[15:0];

    assign LEDR[17:16] = 2'b00;
    assign LEDG = 8'b00000000;       // Không dùng → tắt

    Instruction_Memory IM(
        .read_address(SW[5:0]),
        .instruction(W_INST_OUT)
    );

    // Bạn chưa dùng HEX0-HEX5 → tạm tắt
    assign HEX0 = 7'b1111111;
    assign HEX1 = 7'b1111111;
    assign HEX2 = 7'b1111111;
    assign HEX3 = 7'b1111111;
    assign HEX4 = 7'b1111111;
```

```
Type  ID     Message
             Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 524 warnings
```

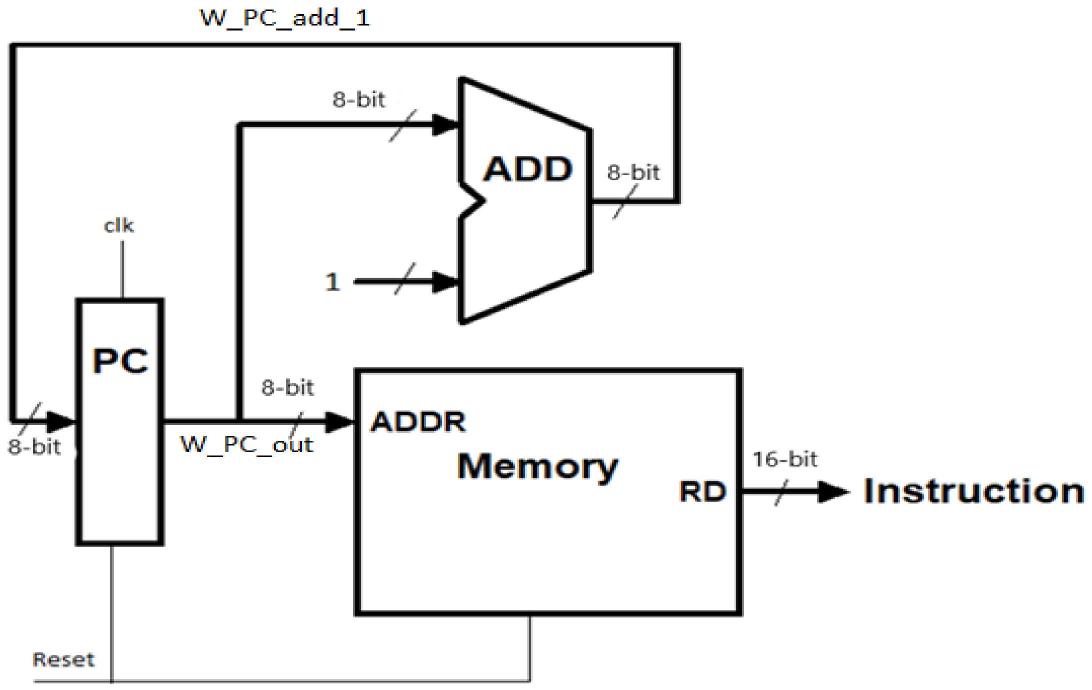
Ex:
Instruction = Imemory[0]
Imemory[0] = 32'b0010000000001000 0000000000100000;

SW[17] = 1 → 16-bit high = 00100000 00001000



III.5 EXPERIMENT NO. 5

III.5.1 AIM: To implement the Instruction memory datapath



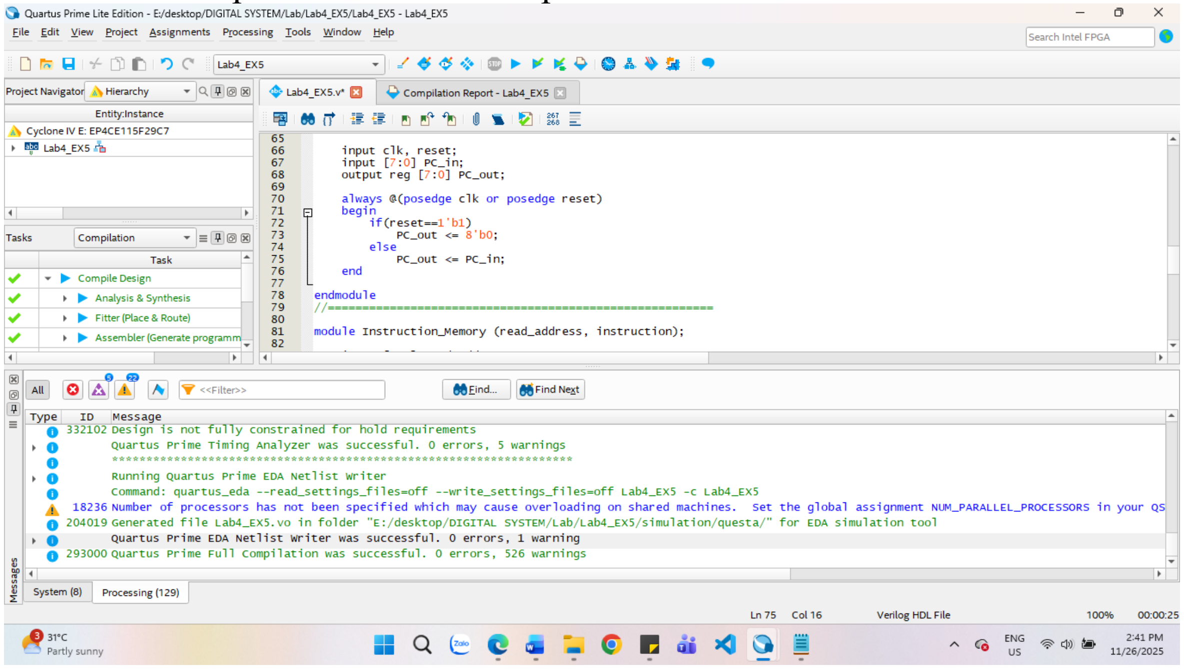
III.5.2 CODE

```
module Instruction_Datapath (clk, reset, Instruction);
        input clk, reset;
        output [15:0] Instruction;
```

endmodule

III.5.3 LAB ASSIGNMENT

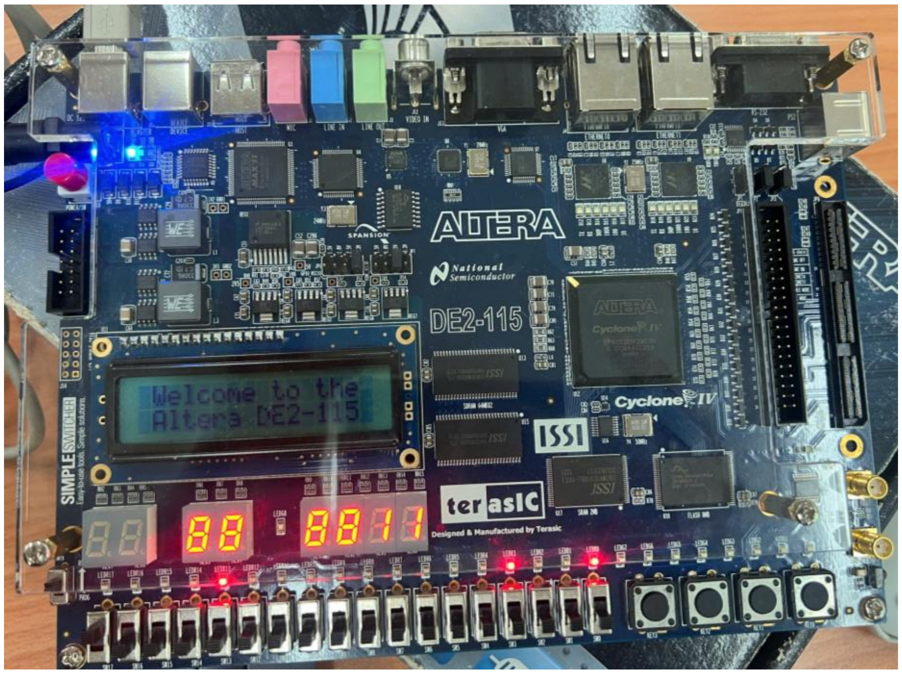
1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT



```
    input clk, reset;
    input [7:0] PC_in;
    output reg [7:0] PC_out;

    always @(posedge clk or posedge reset)
    begin
        if(reset==1'b1)
            PC_out <= 8'b0;
        else
            PC_out <= PC_in;
    end

endmodule
//==============================================================

module Instruction_Memory (read_address, instruction);
```

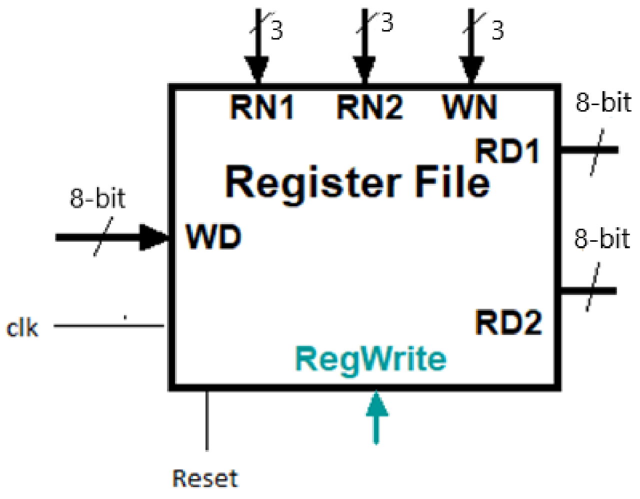
```
Type  ID     Message
      332102 Design is not fully constrained for hold requirements
             Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
             ***************************************************************************
             Running Quartus Prime EDA Netlist Writer
             Command: quartus_eda --read_settings_files=off --write_settings_files=off Lab4_EX5 -c Lab4_EX5
       18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your QS
      204019 Generated file Lab4_EX5.vo in folder "E:/desktop/DIGITAL SYSTEM/Lab/Lab4_EX5/simulation/questa/" for EDA simulation tool
             Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 526 warnings
```

Quartus Prime Lite Edition - E:/desktop/DIGITAL SYSTEM/Lab/Lab4_EX5/Lab4_EX5 - Lab4_EX5
File Edit View Project Assignments Processing Tools Window Help
Lab4_EX5.v*
Compilation Report - Lab4_EX5
Cyclone IV E: EP4CE115F29C7
Lab4_EX5
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
System (8)
Processing (129)
Ln 75 Col 16 Verilog HDL File 100% 00:00:25


Welcome to the
Altera DE2-115
ALTERA
DE2-115
Cyclone IV
terasIC

III.6 EXPERIMENT NO. 6

III.6.1 AIM: To implement the Register File



III.6.2 CODE

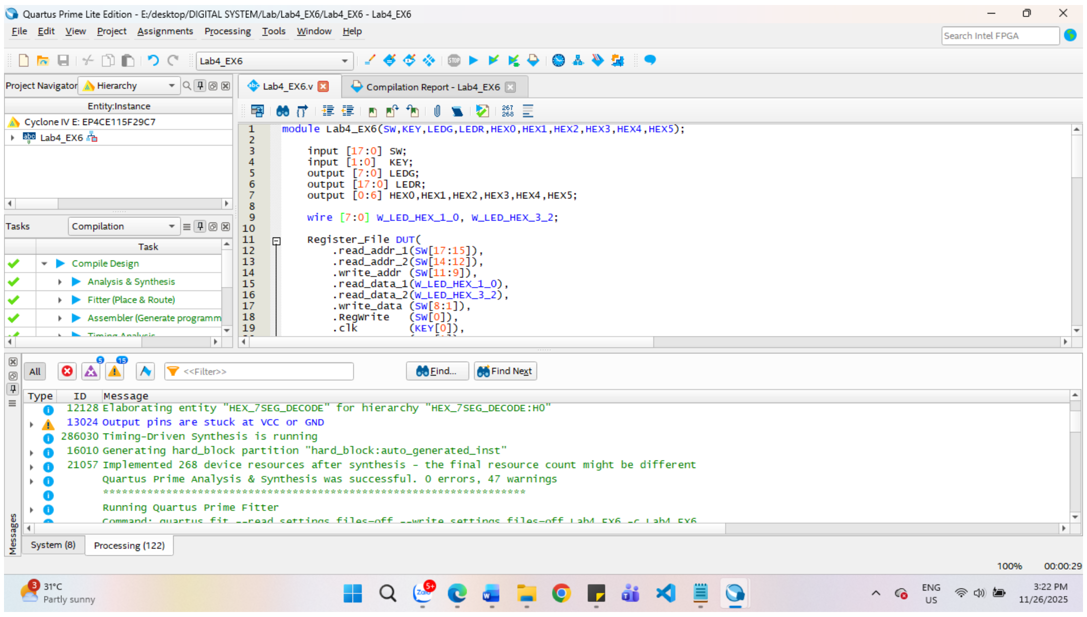
```
module Register_File (read_addr_1, read_addr_2, write_addr, read_data_1, read_data_2,
write_data, RegWrite, clk, reset);
        input [2:0] read_addr_1, read_addr_2, write_addr;
        input [7:0] write_data;
        input clk, reset, RegWrite;
        output [7:0] read_data_1, read_data_2;

        reg [7:0] Regfile [7:0];

        integer k;

        assign read_data_1 = Regfile[read_addr_1];


        assign read_data_2 = Regfile[read_addr_2];

        always @(posedge clk or posedge reset)
        begin
                if (reset==1'b1)
                begin
                        for (k=0; k<8; k=k+1)
                        begin
                                Regfile[k] = 8'b0;
                        end
                end
```

```
            else if (RegWrite == 1'b1) Regfile[write_addr] = write_data;
    end

endmodule
```

reset	clk	read_addr_1	read_data_1	read_addr_2	read_data_2	RegWrite	write_addr	write_data
0	x	x	32'b0	x	32'b0	x	x	x
1	↑	x	x	x	x	0	0x01	0x55
1	↑	x	x	x	x	1	0x01	0x55
1	↑	x	x	x	x	0	0x3	0xAA
1	↑	x	x	x	x	1	0x3	0xAA
1	↑	0x01	0x55	0x03	0xAA	0	x	x
1	↑	x	x	x	x	0	0x04	0x11
1	↑	x	x	x	x	1	0x04	0x11
1	↑	x	x	x	x	0	0x6	0x99
1	↑	x	x	x	x	1	0x6	0x99
1	↑	3'd4	0x11	0x6	0x99	0	x	x

III.6.3 LAB ASSIGNMENT

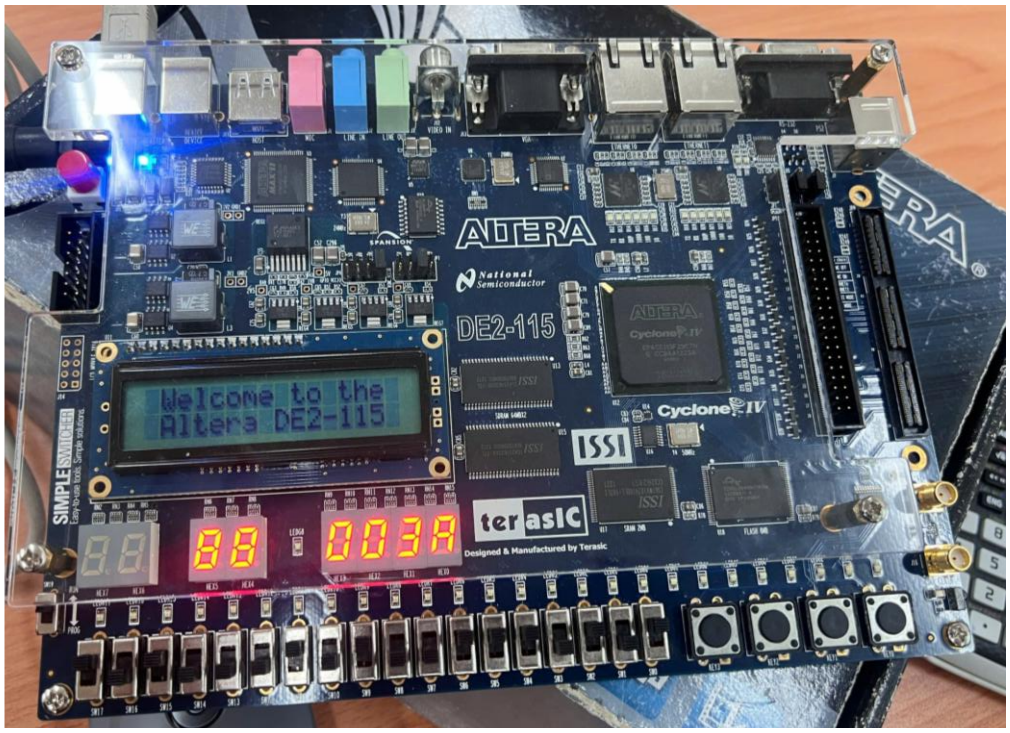
1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT



```
module Lab4_EX6(SW,KEY,LEDG,LEDR,HEX0,HEX1,HEX2,HEX3,HEX4,HEX5);

    input [17:0] SW;
    input [1:0]  KEY;
    output [7:0] LEDG;
    output [17:0] LEDR;
    output [0:6] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5;

    wire [7:0] W_LED_HEX_1_0, W_LED_HEX_3_2;

    Register_File DUT(
        .read_addr_1(SW[17:15]),
        .read_addr_2(SW[14:12]),
        .write_addr (SW[11:9]),
        .read_data_1(W_LED_HEX_1_0),
        .read_data_2(W_LED_HEX_3_2),
        .write_data (SW[8:1]),
        .RegWrite   (SW[0]),
        .clk        (KEY[0]),
```

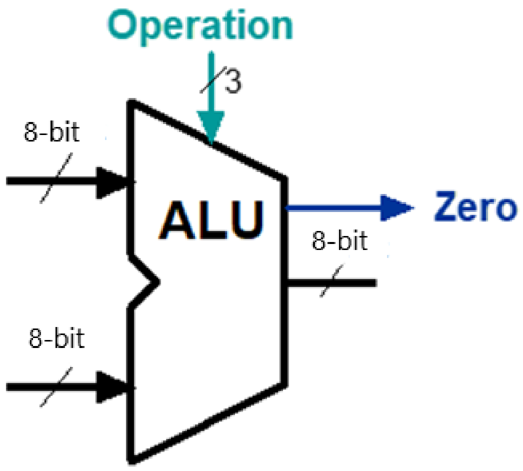
```
Type  ID     Message
      12128 Elaborating entity "HEX_7SEG_DECODE" for hierarchy "HEX_7SEG_DECODE:H0"
      13024 Output pins are stuck at VCC or GND
     286030 Timing-Driven Synthesis is running
      16010 Generating hard_block partition "hard_block:auto_generated_inst"
      21057 Implemented 268 device resources after synthesis - the final resource count might be different
            Quartus Prime Analysis & Synthesis was successful. 0 errors, 47 warnings
            ************************************************************************
            Running Quartus Prime Fitter
```

Quartus Prime Lite Edition - E:/desktop/DIGITAL SYSTEM/Lab/Lab4/Lab4_EX6/Lab4_EX6 - Lab4_EX6
File Edit View Project Assignments Processing Tools Window Help
Lab4_EX6.v
Compilation Report - Lab4_EX6
Cyclone IV E: EP4CE115F29C7
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
System (8)
Processing (122)



III.7 EXPERIMENT NO. 7

III.7.1 AIM: To implement the ALU



III.7.2 CODE

```
module alu(
        input [2:0] alufn,
        input [7:0] ra,
        input [7:0] rb_or_imm,
        output reg [7:0] aluout,
        output reg zero);
```

```
    parameter     ALU_OP_ADD     = 3'b000,
                  ALU_OP_SUB     = 3'b001,
                  ALU_OP_AND     = 3'b010,
                  ALU_OP_OR      = 3'b011,
                  ALU_OP_NOT_A   = 3'b100,
                  ALU_OP_LW      = 3'b101,
                  ALU_OP_SW      = 3'b110,
                  ALU_OP_BEQ     = 3'b111;
    always @(*)
    begin
            case(alufn)
                  ALU_OP_ADD     : aluout = ra + rb_or_imm;
                  ALU_OP_SUB     : aluout = ra - rb_or_imm;
                  ALU_OP_AND     : aluout = ra & rb_or_imm;
                  ALU_OP_OR      : aluout = ra | rb_or_imm;
                  ALU_OP_NOT_A   : aluout = ~ ra;
                  ALU_OP_LW      : aluout = ra + rb_or_imm;
                  ALU_OP_SW      : aluout = ra + rb_or_imm;
                  ALU_OP_BEQ     : begin
                                     zero = (ra==rb_or_imm)?1'b1:1'b0;
                                     aluout = ra - rb_or_imm;
                                   end
            endcase
    end

endmodule
```

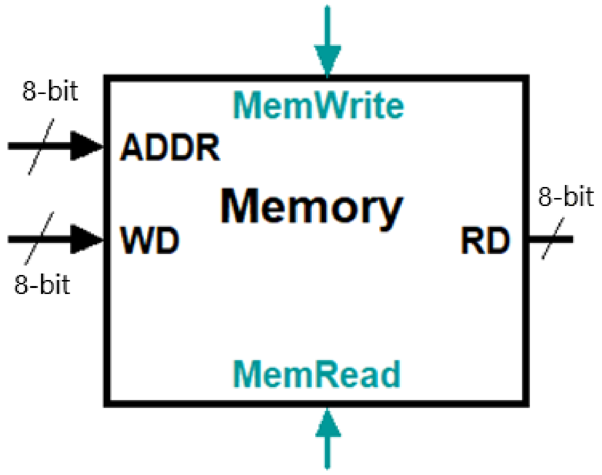
alufn	ra	rb_or_imm	aluout	zero
3'b000	32'd8	32'd2	32'd10	0
3'b001	32'd8	32'd2	32'd6	0
3'b010	32'b1000	32'b0010	32'b0000	0
3'b011	32'b1000	32'b0010	32'b1010	0
3'b100	32'b1000	32'd2	32'b0111	0
3'b101	32'd8	32'd2	32'd10	0
3'b110	32'd8	32'd2	32'd10	0
3'b111	32'd8	32'd2	32'd6	0
3'b111	32'd8	32'd8	32'd0	1

III.7.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT

III.8 EXPERIMENT NO. 8

III.8.1 AIM: To implement the 32 bit RAM

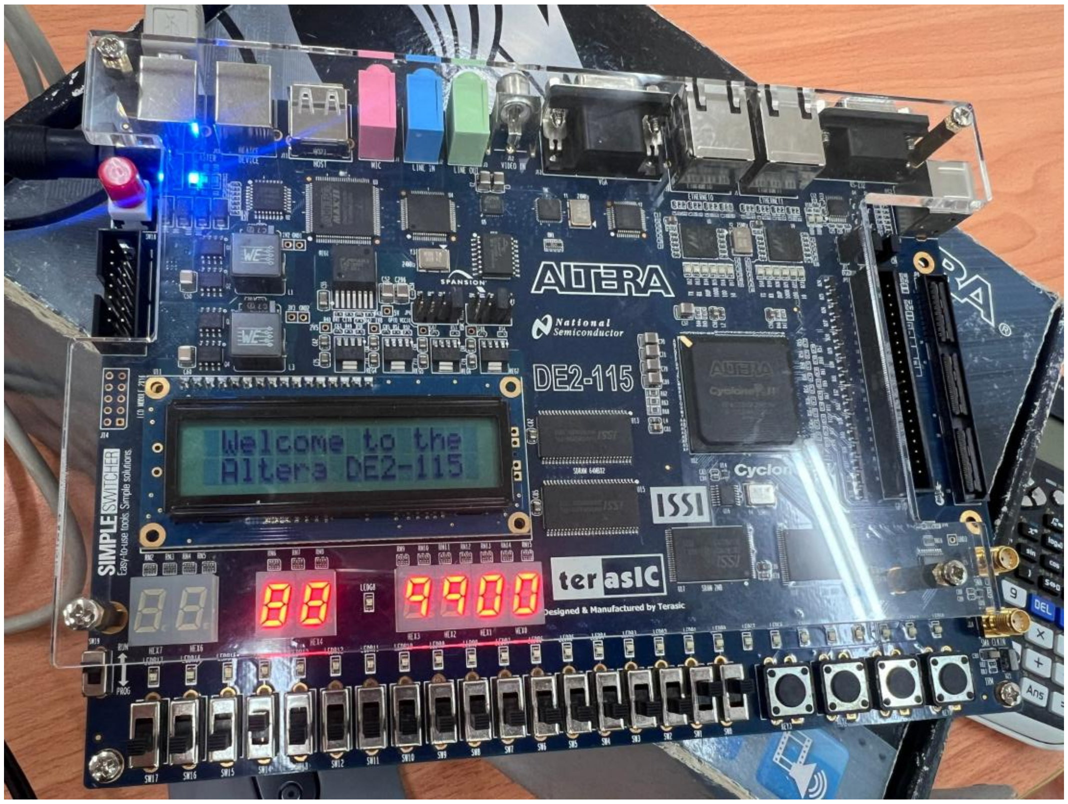


III.8.2 CODE

```
module Data_Memory (addr, write_data, read_data, clk, reset, MemRead, MemWrite);
        input [7:0] addr;
        input [7:0] write_data;
        output [7:0] read_data;
        input clk, reset, MemRead, MemWrite;
        reg [7:0] DMemory [7:0];
        integer k;
        assign read_data = (MemRead) ? DMemory[addr] : 8'bx;




        always @(posedge clk or posedge reset)
        begin
                if (reset == 1'b1)
                        begin
                                for (k=0; k<8; k=k+1)
                                  begin
                                     DMemory[k] = 8'b0;
                                  end
                        end
                else
                        if (MemWrite) DMemory[addr] = write_data;
        end
endmodule
```

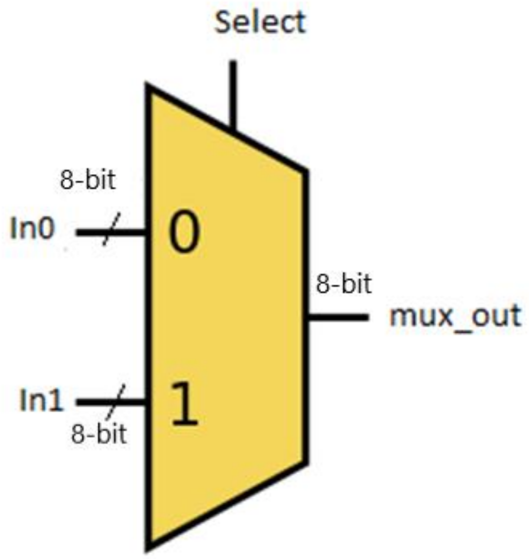
III.8.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT



III.9 EXPERIMENT NO. 9

III.9.1 AIM: To implement Multiplexer



III.9.2 CODE

module Mux_N_bit (in0, in1, mux_out, select);

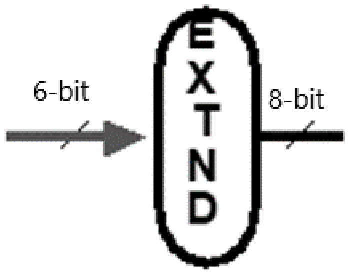
```
        parameter N = 32;
        input [N-1:0] in0, in1;
        output [N-1:0] mux_out;
        input control;
        assign mux_out = select? in1: in0 ;
endmodule
```

III.9.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT

III.10 EXPERIMENT NO. 10

III.10.1 AIM: To implement Sign_Extension



III.10.2 CODE

```
module Sign_Extension (sign_in, sign_out);
        input [15:0] sign_in;
        output [31:0] sign_out;
        assign sign_out[15:0]=sign_in[15:0];
        assign sign_out[31:16]=sign_in[15]?{16{1'b1}}:16'b0;
endmodule
```

III.10.3 LAB ASSIGNMENT

1. Write testbenches to verify above module and attach waveforms.
2. Write the Top level module to implement this module in FPGA KIT

III. LAB REPORT GUIDELINES

Students write up a report on the Verilog HDL implementation experiment projects created in this lab. The lab report should include Verilog code for the module under test, Verilog test bench code and a truth table results, and example data input and output to validate the experiment. Simulation Result in form of Simulation Capture Screen.